

Отчёт по лабораторной работе №3

Дисциплина: Архитектура компьютера

Зарина Исмаилбековна Исаева

Содержание

1	Цель работы	1
2	Задание.....	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы	3
5	Выводы.....	5
6	Список литературы.....	6

1 Цель работы

Цель данной лабораторной работы — освоить процедуры компиляции и сборки программ, написанных на ассемблере NASM.

2 Задание

1. Создание программ Hello world!
2. Работа с транслятором NASM
3. Работ с расширенным синтаксисом командной строки NASM
4. Работа с компоновщиком LD
5. Запуск исполняемого файла
6. Задания для самостоятельной работы

3 Теоретическое введение

Основными функциональными элементами любой ЭВМ являются центральный процессор, память и периферийные устройства. Взаимодействие этих устройств осуществляется через общую шину, к которой они подключены. Физически шина представляет собой большое количество проводников, соединяющих устройства друг с другом. В современных компьютерах проводники выполнены в виде электропроводящих дорожек на материнской плате. Основной задачей процессора является обработка информации, а также организация координации всех узлов

компьютера. В состав центрального процессора входят следующие устройства: - арифметико-логическое устройство (АЛУ) — выполняет логические и арифметические действия, необходимые для обработки информации, хранящейся в памяти; - устройство управления (УУ) — обеспечивает управление и контроль всех устройств компьютера; - регистры — сверхбыстрая оперативная память небольшого объёма, входящая в состав процессора, для временного хранения промежуточных результатов выполнения инструкций; регистры процессора делятся на два типа: регистры общего назначения и специальные регистры. Для того, чтобы писать программы на ассемблере, необходимо знать, какие регистры процессора существуют и как их можно использовать. Большинство команд в программах написанных на ассемблере используют регистры в качестве операндов. Практически все команды представляют собой преобразование данных хранящихся в регистрах процессора, это например пересылка данных между регистрами или между регистрами и памятью, преобразование (арифметические или логические операции) данных хранящихся в регистрах. Доступ к регистрам осуществляется не по адресам, как к основной памяти, а по именам. Каждый регистр процессора архитектуры x86 имеет свое название, состоящее из 2 или 3 букв латинского алфавита. В качестве примера приведем названия основных регистров общего назначения (именно эти регистры чаще всего используются при написании программ): - RAX, RCX, RDX, RBX, RSI, RDI - 64-битные - EAX, ECX, EDX, EBX, ESI, EDI - 32-битные - AX, CX, DX, BX, SI, DI - 16-битные - AH, AL, CH, CL, DH, DL, BH, BL — 8-битные. Другим важным узлом ЭВМ является оперативное запоминающее устройство (ОЗУ). ОЗУ — это быстродействующее энергозависимое запоминающее устройство, которое напрямую взаимодействует с узлами процессора, предназначенное для хранения программ и данных, с которыми процессор непосредственно работает в текущий момент. ОЗУ состоит из одинаковых пронумерованных ячеек памяти. Номер ячейки памяти — это адрес хранящихся в ней данных. Периферийные устройства в составе ЭВМ: - устройства внешней памяти, которые предназначены для долговременного хранения больших объёмов данных. - устройства ввода-вывода, которые обеспечивают взаимодействие ЦП с внешней средой. В основе вычислительного процесса ЭВМ лежит принцип программного управления. Это означает, что компьютер решает поставленную задачу как последовательность действий, записанных в виде программы. Коды команд представляют собой многоразрядные двоичные комбинации из 0 и 1. В коде машинной команды можно выделить две части: операционную и адресную. В операционной части хранится код команды, которую необходимо выполнить. В адресной части хранятся данные или адреса данных, которые участвуют в выполнении данной операции. При выполнении каждой команды процессор выполняет определённую последовательность стандартных действий, которая называется командным циклом процессора. Он заключается в следующем: 1. формирование адреса в памяти очередной команды; 2. считывание кода команды из памяти и её дешифрация; 3. выполнение команды; 4. переход к следующей команде. Язык ассемблера (assembly language, сокращённо asm) — машинно-ориентированный язык низкого уровня. NASM — это открытый проект ассемблера, версии которого доступны под различные операционные системы и который

позволяет получать объектные файлы для этих систем. В NASM используется Intel-синтаксис и поддерживаются инструкции x86-64.

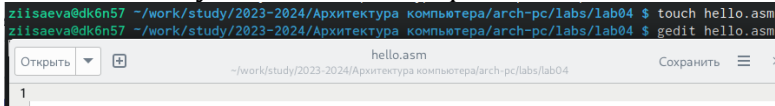
4 Выполнение лабораторной работы

4.1. Создание программы Hello world! С помощью утилиты cd перемещаюсь в каталог, в котором буду работать (рис. [??]).

```
ziisaeva@dk6n57 ~ $ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/labs/lab04
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Создаю в текущем каталоге пустой текстовый файл hello.asm с помощью утилиты

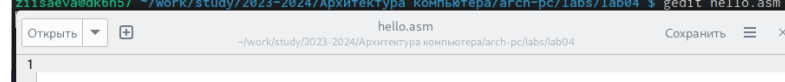
```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ touch hello.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ gedit hello.asm
```



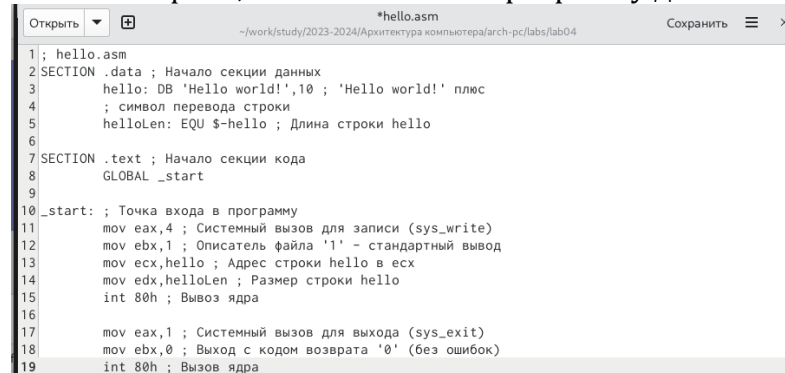
touch (рис. [??]).

Открываю созданный файл в текстовом редакторе (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ touch hello.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ gedit hello.asm
```



Заполняю файл, вставляя в него программу для вывода “Hello world!” (рис. [??]).



```
1; hello.asm
2SECTION .data ; Начало секции данных
3    hello: DB 'Hello world!',10 ; 'Hello world!' плюс
4            ; символ перевода строки
5    helloLen: EQU $-hello ; Длина строки hello
6
7SECTION .text ; Начало секции кода
8    GLOBAL _start
9
10_start: ; Точка входа в программу
11    mov eax,4 ; Системный вызов для записи (sys_write)
12    mov ebx,1 ; Описатель файла '1' - стандартный вывод
13    mov ecx,hello ; Адрес строки hello в ecx
14    mov edx,helloLen ; Размер строки hello
15    int 80h ; Вывод ядра
16
17    mov eax,1 ; Системный вызов для выхода (sys_exit)
18    mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19    int 80h ; Вывод ядра
```

4.2. Превращаю текст программы для вывода “Hello world!” в объектный код с помощью транслятора NASM, используя команду `nasm -f elf hello.asm`, ключ `-f` указывает транслятору `nasm`, что требуется создать бинарный файл в формате ELF (рис. 4.5). Далее проверяю правильность выполнения команды с помощью утилиты `ls`: действительно, создан файл “hello.o” (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf hello.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm hello.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

4.3. Ввожу команду, которая скомпилирует файл `hello.asm` в файл `obj.o`, при этом в файл будут включены символы для отладки (ключ `-g`), также с помощью ключа `-l` будет создан файл листинга `list.lst` (рис. 4.6). Далее проверено с помощью утилиты `ls` правильность выполнения команды (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -o obj.o -f elf -g -l list.lst hello.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm hello.o list.lst obj.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

4.4. Работа с компоновщиком LD Передаю объектный файл hello.o на обработку компоновщику LD, чтобы получить исполняемый файл hello (рис. 4.7). Ключ -o задает имя создаваемого исполняемого файла. Далее проверяю с помощью утилиты ls правильность выполнения команды (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 hello.o -o hello
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst obj.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Выполняю следующую команду (рис. 4.8). Исполняемый файл будет иметь имя main, т.к. после ключа -o было задано значение main. Объективный файл, из которого собран этот исполняемый файл, имеет имя obj.o (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 obj.o -o main
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o list.lst main obj.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

4.5. Запуск исполняемого файла Запускаю на выполнение созданный исполняемый файл hello (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./hello
Hello world!
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

4.6. Задания для самостоятельной работы С помощью утилиты cp создаю в текущем каталоге копию файла hello.as с именем lab4.asm (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ cp hello.asm lab4.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ gedit lab4.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

С помощью текстового редактора gedit открываю файл lab4.asm и вношу изменения в программу так, чтобы она выводила мои имя и фамилию (рис. [??]).

```
Открыть  + lab4.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04

1 ; lab4.asm
2 SECTION .data ; Начало секции данных
3     lab4: DB 'Zarina Isaeva',10
4
5     lab4Len: EQU $-lab4 ; Длина строки lab4
6
7 SECTION .text ; Начало секции кода
8     GLOBAL _start
9
10 _start: ; Точка входа в программу
11     mov eax,4 ; Системный вызов для записи (sys_write)
12     mov ebx,1 ; Описатель файла '1' - стандартный вывод
13     mov ecx,lab4 ; Адрес строки lab4 в есх
14     mov edx,lab4Len ; Размер строки lab
15     int 80h ; Вывоз ядра
16
17     mov eax,1 ; Системный вызов для выхода (sys_exit)
18     mov ebx,0 ; Выход с кодом возврата '0' (без ошибок)
19     int 80h ; Вызов ядра
```

Компилирую текст программ в объективный файл (рис. 4.12). Проверяю с помощью утилиты ls, что файл lab4.o создан (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ nasm -f elf lab4.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o lab4.asm lab4.o list.lst main obj.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Передаю объективный файл lab4.o на обработку компоновщику LD, чтобы получить исполняемый файл lab4 (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ld -m elf_i386 lab4.o -o lab4
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Запускаю исполняемый айл lab4, на экране действительно выводятся мои имя и

фамилия (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ./lab4
Zarina Isaeva
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

К сожалению, я начала работу не в том каталоге, поэтому создаю другую директорию lab4 с помощью mkdir, прописывая полный путь к каталогу, в котором хочу создать эту директорию. Далее копирую из текущего каталога файлы, созданные в процессе выполнения лабораторной работы, с помощью утилиты cp, указывая вместо имени файла символ *, чтобы скопировать все файлы. Команда проигнорирует директории в этом каталоге, т. к. не указан ключ -r, это мне и нужно (рис. 4.15). Проверяю с помощью утилиты ls правильность выполнения команды

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ mkdir ~/work/study/2023-2024/Архитектура\ компь
тера/arch-pc/lab04
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ cp * ~/work/study/2023-2024/Архитектура\ компь
тера/arch-pc/lab04
cp: не указан -r; пропускается каталог 'presentation'
cp: не указан -r; пропускается каталог 'report'
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls ~/work/study/2023-2024/Архитектура\ компьте
ра/arch-pc/lab04
hello hello.asm hello.o lab4 lab4.asm lab4.o list.lst main obj.o
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

(рис. [??]).

Удаляю лишние файлы в текущем каталоге с помощью утилиты rm,ведь копии файлов в другой директории (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ rm hello hello.o lab4 lab4.o list.lst main obj.
o
rm: невозможно удалить 'hello': Нет такого файла или каталога
rm: невозможно удалить 'hello.o': Нет такого файла или каталога
rm: невозможно удалить 'lab4': Нет такого файла или каталога
rm: невозможно удалить 'lab4.o': Нет такого файла или каталога
rm: невозможно удалить 'main': Нет такого файла или каталога
rm: невозможно удалить 'obj.o': Нет такого файла или каталога
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ ls
hello.asm lab4.asm presentation report
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

С помощью команд git add . и git commit добавляю файлы на GitHub, комментируя действие как добавление файлов для лабораторной работы №4 (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git add .
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git commit -m "Add fales for lab04"
[master 2a280c7] Add fales for lab04
2 files changed, 38 insertions(+)
create mode 100644 labs/lab04/hello.asm
create mode 100644 labs/lab04/lab4.asm
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

Отправляю файлы на сервер с помощью команды git push (рис. [??]).

```
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $ git push
Перечисление объектов: 9, готово.
Подсчет объектов: 100% (9/9), готово.
При сжатии изменений используется до 6 потоков
Сжатие объектов: 100% (6/6), готово.
Запись объектов: 100% (6/6), 1.06 КиБ | 1.06 МиБ/с, готово.
Всего 6 (изменений 3), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To github.com:ziisaeva/study_2023-2024_arh-pc.git
  1a8aa2a..2a280c7 master -> master
ziisaeva@dk6n57 ~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab04 $
```

5 Выводы

При выполнении данной лабораторной работы я освоила процедуры компиляции и сборки программ, написанные на ассемблере NASM.

6 Список литературы

1. https://esystem.rudn.ru/pluginfile.php/2089084/mod_resource/content/0/%D0%9B%D0%B0%D0%B1%D0%BE%D1%80%D0%B0%D1%82%D0%BE%D1%80%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20%E2%84%964.%20%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%B8%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%20%D0%BD%D0%B0%20%D1%8F%D0%B7%D1%8B%D0%BA%D0%B5%20%D0%B0%D1%81%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80%D0%B0%20NASM.pdf
2. <https://esystem.rudn.ru/mod/page/view.php?id=1030505>